

# 1. Date, informații, cunoștințe

Auzim adesea vorbindu-se despre “Era informațiilor” sau “societate informațională” sau “tehnologia informației” însă de multe ori cuvântul "informație" este folosit fără a înțelege clar sensul acestui cuvânt, diferența dintre date, informații, cunoștințe.

În general, conținutul gândirii umane operează cu următoarele concepte:

1. **Date** – constau în material brut, fapte, simboluri, numere, cuvinte, poze fără un înțeles de sine stătător, neintegrate într-un context, fără relații cu alte date sau obiecte. Ele se pot obține în urma unor experimente, sondaje etc.
2. **Informații** – prin prelucrarea datelor și găsirea relațiilor dintre acestea se obțin informații care au un înțeles și sunt integrate într-un context. Datele organizate și prezentate într-un mod sistematic pentru a sublinia sensul acestor date devin informații. Pe scurt informațiile sunt date prelucrate. Informațiile se prezintă sub formă de rapoarte, statistici, diagrame etc.
3. **Cunoștințele** sunt colecții de date, informații, adevăruri și principii învățate, acumulate de-a lungul timpului. Informațiile despre un subiect reținute și înțelese și care pot fi folosite în luarea de decizii, formează judecăți și opinii devin cunoștințe. Cu alte cuvinte, cunoștințele apar în momentul utilizării informației .

## 2. Colectarea și analizarea datelor. Modelul conceptual

Primul pas în realizarea unei aplicații de baze de date este analiza datelor și realizarea unei **scheme conceptuale (model conceptual)** al acestor date.

În această etapă sunt analizate natura și modul de utilizare a datelor. Sunt identificate datele care vor trebui memorate și procesate, se împart aceste date în grupuri logice și se identifică relațiile care există între aceste grupuri.

Analiza datelor este un proces uneori dificil, care necesită mult timp, însă este o etapă absolut obligatorie. Fără o analiză atentă a datelor și a modului de utilizare a acestora, vom realiza o bază de date care putem constata în final că nu întrunește cerințele beneficiarului. Costurile modificării acestei baze de date este mult mai mare decât costurile pe care le-ar fi implicat etapa de analiză și realizare a modelului conceptual. Modificarea modelului conceptual este mult mai ușoară decât modificarea unor tabele deja existente, care eventual conțin și o mulțime de date. Ideea de bază a analizei datelor și construirii modelului conceptual este "să măsoari de două ori și să tai o singură dată".

Informațiile necesare realizării modelului conceptual se obțin folosind metode convenționale precum interviuarea oamenilor din cadrul organizației și studierea documentelor folosite.

Odată obținute aceste informații ele trebuiesc reprezentate într-o formă convențională care să poată fi ușor înțeleasă de toată lumea. O astfel de reprezentare este **diagrama entități-relații**, numită și **harta relațiilor**, sau **ERD-ul** (**Entity Relationship Diagram**). Aceste scheme sunt un instrument util care ușurează comunicarea dintre specialiștii care proiectează bazele de date și programatori pe de o parte și beneficiari, pe de altă parte. Aceștia din urmă pot înțelege cu ușurință o astfel de schemă, chiar dacă nu sunt cunoscători în domeniul IT.

În concluzie putem sublinia câteva caracteristici ale ERD-urilor:

- sunt un instrument de proiectare
- sunt o reprezentare grafică a unui sistem de date
- oferă un model conceptual de înalt nivel al bazelor de date
- sprijină înțelegerea de către utilizatori a datelor și a relațiilor dintre acestea
- sunt independente de implementare.

În cele ce urmează vom prezenta principalele elemente care intră în componența unui ERD precum și convențiile de reprezentare a acestora.

### 3. Entități. Instanțe. Atribute. Identificator unic.

O **entitate** este un lucru, obiect, persoană sau eveniment care are semnificație pentru afacerea modelată, despre care trebuie să colectăm și să memorăm date. O entitate poate fi un lucru real, tangibil precum o clădire, o persoană, poate fi o activitate precum o programare sau o operație, sau poate fi o noțiune abstractă.

O entitate este reprezentată în ERD printr-un dreptunghi cu colțurile rotunjite. Numele entității este întotdeauna un *substantiv la singular* și se scrie în partea de sus a dreptunghiului cu *majuscule*, ca în figura I.1.1.



O entitate este de fapt o clasă de obiecte și pentru orice entitate există mai multe *instanțe* ale sale. O instanță a unei entități este un obiect, persoană, eveniment, particular din clasa de obiecte care formează entitatea. De exemplu, elevul **X** din clasa a IX-a A de la Liceul de Informatică din localitatea **Y** este o instanță a entității **ELEV**.

După cum se vede pentru a preciza o instanță a unei entități, trebuie să specificăm unele caracteristici ale acestui obiect, să-l descriem (precizăm de exemplu numele, clasa, școala etc). Așadar, după ce am identificat entitățile trebuie să descriem aceste entități în termeni reali, adică să le stabilim *atributele*. Un atribut este orice detaliu care servește la identificarea, clasificarea, cuantificarea, sau exprimarea stării unei instanțe a unei entități. Atributele sunt informații specifice ce trebuie cunoscute și memorate.



De exemplu atributele entității **ELEV** sunt nume, prenume, adresa, număr de telefon, adresa de email, data nașterii etc.

În cadrul unui ERD, atributele se vor scrie imediat sub numele entității, cu litere mici. Un atribut este un *substantiv la singular* (vezi figura I.1.2).

Un atribut poate fi *obligatoriu* sau *opțional*. Dacă un atribut este obligatoriu, pentru fiecare instanță a entității respective trebuie să avem o valoare pentru acel atribut, de exemplu este obligatoriu să cunoaștem numele elevilor. Pentru un atribut opțional putem avea instanțe pentru care nu cunoaștem valoarea atributului respectiv. De exemplu atributul **email** al entității **ELEV** este opțional, un elev putând să nu aibă adresă de email. Un atribut obligatoriu este precedat în ERD de un asterisc \*, iar un atribut opțional va fi precedat de un cerculeț o.

Atributele care definesc în mod unic instanțele unei entități se numesc *identificator unic* (UID). UID-ul unei entități poate fi compus dintr-un singur atribut, de exemplu codul numeric personal poate fi un identificator unic pentru entitatea **ELEV**. În alte situații, identificatorul unic este compus dintr-o combinație de două sau mai multe atribute. De exemplu combinația dintre titlu, numele autorului și data apariției poate forma unicul identificator al entității **CARTE**. Oare

combinația titlu și nume autor nu era suficientă? Răspunsul este NU, deoarece pot exista de exemplu mai multe volume scrise de Mihai Eminescu având toate titlul Poezii, dar apărute la date diferite.



Atributele care fac parte din identificatorul unic al unei entități vor fi precedate de semnul diez # (figura I.1.2 și I.1.3). **Atributele din UID sunt întotdeauna obligatorii**, însă semnul # este suficient, nu mai trebuie pus și un semn asterisc în fața acestor atribute.

Valorile unor atribute se pot modifica foarte des, ca de exemplu atributul vârstă. Spunem în acest caz că avem de a face cu un **atribut volatil**. Dacă valoarea unui atribut însă se modifică foarte rar sau deloc (de exemplu data nașterii) acesta este un atribut **non-volatil**. Evident este de preferat să folosim atribute non-volatile atunci când acest lucru este posibil.

## 4. Relații între entități

În lumea reală, obiectele nu există izolat. Între ele există relații. Așadar, după ce ați identificat care sunt entitățile și atributele acestor entități este timpul să punem în evidență relațiile care există între aceste entități, modul în care acestea comunică între ele. O **relație** este o asociere, legătură, sau conexiune existentă între entități și care are o semnificație pentru afacerea modelată. Orice relație este bidirecțională, legând două entități sau o entitate cu ea însăși. De exemplu, elevii studiază mai multe materii, o materie e studiată de către elevi.

Orice relație este caracterizată de următoarele elemente:

- 1. numele relației ;      2. opționalitatea relației;      3. gradul (cardinalitatea) relației.

Să luăm de exemplu relația existentă între entitățile **JUCĂTOR** și **ECHIPĂ**. Vom spune:

Un **JUCĂTOR** joacă într-o **ECHIPĂ**. Și La o **ECHIPĂ** trebuie să joace unul sau mai mulți **JUCĂTORI**.

- **Numele relației** este: *joacă*.

- Pentru a stabili **opționalitatea** relației trebuie să răspundem la următoarea întrebare: Un jucător **trebuie** să joace într-o echipă? Se poate ca un jucător să nu joace în nici o echipă? Dacă acceptăm că toți jucătorii trebuie să joace într-o echipă relația este obligatorie sau mandatorie și vom spune: Un **JUCĂTOR trebuie** să joace într-o **ECHIPĂ**.

Dacă însă acceptăm că există jucători care nu joacă în nici o echipă (de exemplu li s-a terminat contractul și în momentul de față nu mai joacă la nici o echipă), atunci relația este opțională.

În acest caz vom spune: Un **JUCĂTOR poate** juca la o **ECHIPĂ**.

- **Cardinalitatea** relației este dată de numărul de instanțe ale entității din partea dreaptă a relației care pot intra în relație cu o instanță a entității din partea stângă a relației. Adică va trebui să răspundem la întrebări de genul: La câte echipe poate juca un jucător? Răspunsurile posibile sunt **unul și numai unul**, sau **unul sau mai mulți**. Vom spune:

Un **JUCĂTOR** trebuie/poate să joace la o **ECHIPĂ și numai una**.

sau Un **JUCĂTOR** trebuie/poate să joace la **una sau mai multe ECHIBE**.

Cea mai realistă varietate a relației este așadar:

Un **JUCĂTOR** poate să joace la o **ECHIPĂ** și numai una.

## 4. Convenții de reprezentare a relațiilor

În cadrul diagramei entități-relații, o relație va fi reprezentată printr-o linie ce unește cele două entități.

Deoarece o relație este bidirecțională, linia ce unește cele două entități este compusă din două segmente distincte, câte una pentru fiecare entitate. Tipul segmentului ce pleacă de la o entitate ne va indica **opționalitatea** relației dintre această entitate și entitatea aflată în cealaltă parte a relației. Dacă acest segment este continuu este vorba de o relație obligatorie, o linie întreruptă indică o relație opțională.





De exemplu în figura I.1.4 segmentul ce pleacă de la entitatea JUCĂTOR fiind **întreruptă** înseamnă că un jucător **poate** juca la o echipă, adică relația este opțională. Segmentul ce pleacă dinspre entitatea ECHIPĂ este **continuu**, deci la o echipă **trebuie** să joace jucători.



**Figura I.1.4.** Reprezentarea relațiilor

Modul în care o linie se termină spre o entitate este important. Dacă se termină printr-o linie simplă, înseamnă că o instanță și numai una a acestei entități este în relație cu o instanță a celeilalte entități. În exemplul anterior, linia de la **JUCĂTOR** la **ECHIPĂ** se termină în partea dinspre **ECHIPĂ** cu o *linie simplă*, deci un jucător joacă la *o* echipă *și numai una*.

Dacă linia se termină cu trei linii (picior de cioară) înseamnă că mai multe instanțe ale entității pot corespunde unei instanțe a celeilalte entități. În exemplul anterior linia de la **ECHIPĂ** la **JUCĂTOR** se termină cu *piciorul de cioară*, înseamnă că unei instanțe a entității **ECHIPĂ** îi corespund mai multe instanțe ale entității **JUCĂTOR**, adică o echipă are *unul sau mai mulți* jucători.

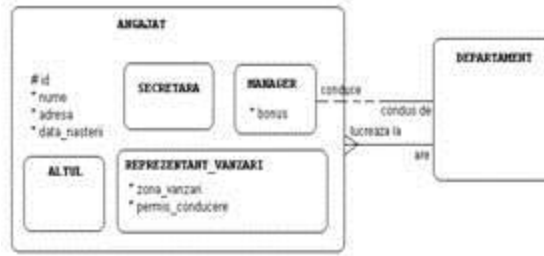
Caracteristica relației	Valoare	Mod de reprezentare
<b>Numele relației</b>	un verb	se scrie deasupra relației
<b>Opționalitatea</b>	relație obligatorie (TREBUIE)	linie continuă 
	relație opțională (POATE)	linie întreruptă 
<b>Cardinalitatea</b>	una și numai una	linie simplă 
	una sau mai multe	picior de cioară 

## Tipuri și subtipuri

În lumea reală obiectele sunt de obicei clasificate. Astfel vorbim despre animale vertebrate și nevertebrate, despre licee teoretice, colegii, grupuri școlare etc. E normal ca în modelarea bazelor de date să putem modela și astfel de clasificări.

Un **subtip** sau o **subentitate** este o clasificare a unei entități care are caracteristici comune cu entitatea generală, precum atribute și relații. Subtipurile se reprezintă în cadrul hărții relațiilor ca entități în interiorul altei entități. Atributele și relațiile comune tuturor subtipurilor se vor reprezenta la nivelul **supertipului**, sau **superentității**. Atributele și relațiile supertipului vor fi moștenite de către subtipuri.

Un subtip poate avea la rândul său alte subtipuri incluse.



**Figura I.4.1.** Folosirea subtipurilor și supertipurilor

Subtipurile trebuie să respecte două reguli importante:

- trebuie să acopere toate cazurile posibile de instanțe ale supertipului, cu alte cuvinte, orice instanță a supertipului trebuie să aparțină unui subtip. De multe ori ERD-urile includ un subtip "ALTUL" pentru a acoperi toate situațiile, și pentru a permite viitoare dezvoltări ale modelului.

subtipurile trebuie să se excludă reciproc. Această regulă se traduce pe exemplul de mai sus în faptul că un angajat nu poate fi, de exemplu, și manager și secretară în același timp.

## Documentare Business Rules

Pentru ca modelul conceptual să fie complet se definesc **reguli structurale** (-indica tipuri de informații ce vor fi stocate și cum relaționează ele) și reguli procedurale (legate de timp, etc, -acestea nu se reprezintă pe ERD, ci trebuie implementate în programare).

## Tipuri de relații

Variantele de relații ce pot exista între două entități sunt prezentate mai jos:

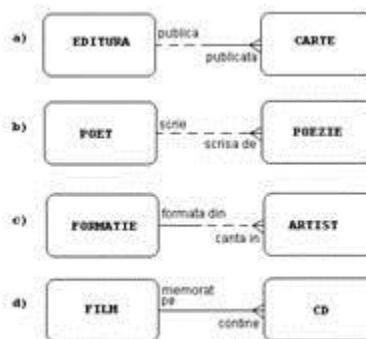
- **relații one-to-one** – acest tip de relație este destul de rar întâlnit. Uneori astfel de relații pot fi modelate transformând una dintre entități în atribut al celeilalte entități.



**Figura I.1.5.** Relații one-to-one

- **relații one-to-many** – sunt cele mai întâlnite tipuri de relații, însă și aici cazurile c și d prezentate în figura I.1.6 sunt mai puțin uzuale.

Să facem câteva observații pe marginea exemplelor din figura I.1.6. **Cazul a** este foarte des întâlnit. La **cazul b**, am ales o relație opțională dinspre **POEZIE** spre **POET** deoarece poate fi vorba de o poezie populară și în acest caz nu există un poet cunoscut. La **cazul c**, am considerat că o formație nu poate exista fără a avea cel puțin un membru, însă un artist poate avea o carieră solo, deci nu face parte din nici o formație. **Varianta d** modelează o colecție de filme memorate pe CD-uri. Pentru afacerea considerată, un CD conține obligatoriu un film, dar unul singur, însă un film poate să nu încapă pe un singur CD de aceea el este poate fi memorat pe unul sau mai multe CD-uri.

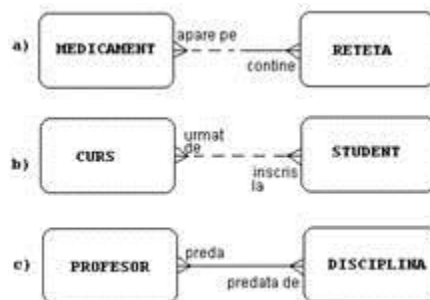


**Figura I.1.6.** Relații one-to-many

- **relații many-to-many** – aceste tipuri de relații apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate. Figura I.1.7 prezintă câteva exemple de relații many-to-many. La punctul b am considerat că un curs poate apărea pe oferta de cursuri a unei facultăți, însă poate să nu fie aleasă de nici un student de aceea un curs **poate** fi urmat de unul sau mai mulți studenți. Invers, este posibil ca un student să fi terminat studiile și să



se pregătească pentru susținerea examenului de licență și de aceea el nu mai frecventează nici un curs. La punctul c, un profesor angajat al unei școli **trebuie** să predea cel puțin o disciplină. Iar o disciplină din planul de învățământ trebuie să fie predată de cel puțin un profesor.

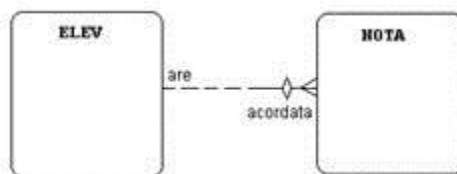


**Figura I.1.7.** Relații many-to-many

## Transferabilitate

Spunem că o relație este nontransferabilă dacă o asociație între două instanțe ale celor două entități, odată stabilită, nu mai poate fi modificată. Nontransferabilitatea unei relații se reduce la faptul că valorile cheii străine corespunzătoare relației respective nu pot fi modificate. Condiția de nontransferabilitate a unei relații este asigurată prin program. De aceea trebuie să documentăm această restricție. În ERD o relație nontransferabilă se notează cu un romb pe linia corespunzătoare relației, înspre entitatea a cărei cheie străină nu este permis să o modificăm (adică în partea cu many a unei relații one-to-many).

În figura I.4.5 este dat un exemplu de relație nontransferabilă. Este vorba despre notele date elevilor. Este normal ca o notă dată unui elev să nu poată fi apoi transferată unui alt elev.

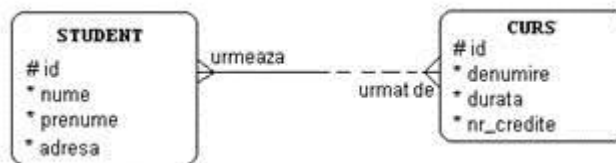


**Figura I.4.5.** Relații nontransferabile

# Rezolvarea relațiilor many-to-many

După cum am precizat mai devreme relațiile many-to-many pot apărea într-o primă fază a proiectării bazei de date însă ele nu au voie să apară în schema finală. Să considerăm relația din figura I.1.14 dintre entitățile STUDENT și CURS. Se știe că orice curs se termină în general cu un examen. Unde vom memora nota studentului la fiecare examen?

Figura I.1.14



Dacă încercăm să introducem atributul **NOTA** la entitatea **STUDENT**, nu vom ști cărei materii corespunde acea notă, întrucât unei instanțe a entității student îi corespund mai multe instanțe ale entității **CURS**. Invers dacă încercăm să memorăm nota în cadrul entității **CURS**, nu vom ști cui student îi aparține acea notă.

Rezolvarea unei relații many-to-many constă introducerea unei noi entități numită **entitate de intersecție**, pe care o legăm de entitățile originale prin câte o relație one-to-many.

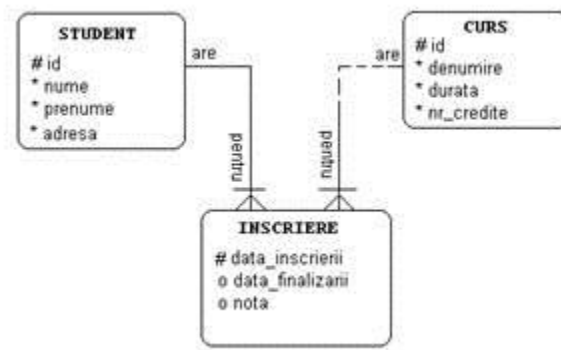
Pașii în rezolvarea unei relații many-to-many sunt următorii:

- 1) se găsește entitatea de intersecție, pentru exemplul nostru vom introduce entitate **INSCRIERE**.
- 2) crearea noilor relații
  - opționalitatea: relațiile care pleacă **din entitatea de intersecție sunt întotdeauna obligatorii în această parte**. În partea dinspre entitățile originale, relațiile vor păstra opționalitatea relațiilor inițiale.
  - cardinalitatea: ambele relații sunt de tip one-to-many, iar partea cu many va fi întotdeauna înspre entitatea de intersecție.
  - numele noilor relații
- 3) adăugarea de attribute în cadrul entității de intersecție, dacă acestea există. În exemplul nostru ne poate interesa de exemplu data la care s-a înscris un student la un curs, data la care a finalizat cursul precum și nota obținută la sfârșitul cursului.

- 4) stabilirea identificatorului unic pentru entitatea de intersecție: dacă entitatea de intersecție nu are un identificator unic propriu, atunci acesta se poate forma din identificatorii unici ai entităților inițiale la care putem adăuga attribute ale entității de intersecție.

În exemplul nostru, identificatorul unic al entității de intersecție este format din id-ul studentului, id-ul cursului și data înscrierii la curs.

- 5) Faptul că identificatorul unic al unei entități preia identificatorul unic din altă entitate cu care este legată este reprezentat grafic prin bararea relației respective, înspre entitatea care preia UID-ul celeilalte entități.



**Analiza CRUD**-se referă la CREATE, RETRIVE, UPDATE, DELETE-(creare, regăsire, actualizare, stergere) operații ce fac din ERD un model complet. Se verifică dacă modelul exprimă toate operațiile ce se pot face și nu are elemente inutile, etc.

## UID artificial și compus

**UID**-(Unique Identifier)-e atributul ce identifica in mod unic entitatea(ex: CNP, cod, id,). Daca e nevoie de o combinație de mai multe attribute care sa identifice in mod unic entitatea , e vorba de un UID compus. Daca se recurge la o modalitate de identificare printr-un cod artificial oferit in mod automat de program, e vorba de UID artificial.

## Ce este normalizarea?

Normalizarea este o tehnică de proiectare a bazelor de date prin care se elimină (sau se evită) anumite anomalii și inconsistențe a datelor. O baza de date bine proiectată nu permite astfel ca datele să fie redundante, adică aceeași informație să se găsească în locuri diferite, sau să memorezi în baza de date, informații care se pot deduce pe baza altor informații memorate în aceeași bază de date. Anomaliile care pot să apară la o bază de date nenormalizată sunt următoarele:

- **anomalii la actualizarea** datelor la o bibliotecă se înregistrează într-o tabelă următoarele date despre cărți: ISBN, titlu, autor, preț, subiect, editura, adresa editurii. La un moment dat o editură își schimbă adresa. Bibliotecara va trebui să modifice adresa editurii respective, în înregistrările corespunzătoare tuturor cărților din bibliotecă apărute la respectiva editură. Dacă această modificare nu se face cu succes, unele dintre înregistrări rămânând cu vechea adresă, apare din nou o inconsistență a datelor.

- **anomalii de inserare** – în exemplul anterior, nu vom putea memora adresa unei edituri, lucru inacceptabil dacă dorim să avem informații și despre edituri a căror cărți nu le avem în bibliotecă, eventual de la care dorim să facem comenzi.

- **anomalii de ștergere** – să presupunem că într-o tabelă memorăm următoarele informații: codul studentului, codul cursului, codul profesorului. La un moment dat, nici un student nu mai dorește să participe la un anumit curs. Ștergând toate înregistrările corespunzătoare cursului, nu vom mai putea ști niciodată cine predă acel curs.

Edgar Codd a definit primele trei forme normale 1NF, 2NF și 3NF. Ulterior s-au mai definit formele normale 4NF, 5NF, 6NF care însă sunt rar folosite în proiectarea bazelor de date.

## Prima formă normală

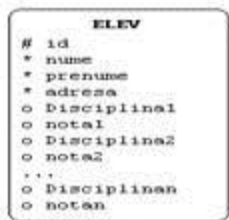
**O entitate se găsește în prima formă normală dacă și numai dacă:- nu există attribute cu valori multiple;- nu există attribute sau grupuri de attribute care se repetă.** Cu alte cuvinte toate attributele trebuie să fie atomice, adică să conțină o singură informație.

Dacă un atribut are valori multiple, sau un grup de attribute se repetă, atunci trebuie să creați o entitate suplimentară pe care să o legați de entitatea originală printr-o relație de 1:m. În noua entitate vor fi introduse attributele sau grupurile de attribute care se repetă.

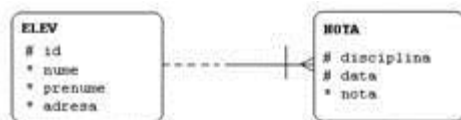
Să considerăm entitatea din figura I.2.1, referitoare la notele elevilor unei clase. Câteva observații referitoare la această entitate: câte discipline are un elev? Câte perechi (disciplina, nota) va trebui să aibă entitatea Elevi? Să spunem că știm exact câte discipline maxim poate studia un elev. Ce se întâmplă dacă în anul viitor școlar acest număr de discipline va fi mai mare? În plus, la o materie un elev poate avea mai multe note. Câte note? Cum memorăm aceste note? Le punem în câmpul corespunzător disciplinei cu virgulă între ele?

Cum rezolvăm această problemă? Vom crea o nouă entitate în care vom introduce disciplina și nota la disciplina respectivă (vezi figura I.2.2.).

În acest fel fiecărui elev îi pot corespunde oricâte note, iar la o disciplină poate avea oricâte note, singura restricție conform acestui model fiind că un elev nu va putea primi în aceeași zi la aceeași materie mai multe note.



**Figura I.2.1.**



**Figura I.2.2**

Un alt exemplu de încălcare a regulilor primei forme normale, puțin mai "ascuns", este prezentat în figura I.2.5. De ce? Pentru că adresa este de forma "str. Florilor, bl. 45, sc. A, ap. 28, etaj 3, Brașov, cod 123123", formă care de fapt conține mai multe informații elementare. Așadar, în mod normal acest atribut ar trebui "spart" în mai multe attribute ca în figura I.2.6.



**Figura I.2.5**

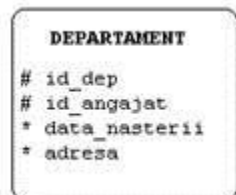


**Figura I.2.6.**

Noile attributele introduse sunt opționale întrucât dacă elevul locuiește la casă, probabil attributele bloc, apartament, scara, etaj, nu au sens. Invers dacă elevul locuiește la bloc, probabil nu poate fi completat numărul.

Pentru acest tip de încălcare a regulilor forme normale 1NF poate fi totuși ignorată, decizia depinzând de natura fenomenului, sau afacerii modelate. În exemplul anterior, întrucât datele din interiorul unei adrese este puțin probabil să se modifice, modificându-se el mult adresa completă a unui elev, se poate decide să nu operăm modificarea anterioară. Dacă însă aceste informații s-ar modifica frecvent, de exemplu denumirile străzilor s-ar modifica mereu, atunci probabil modificarea este de dorit.

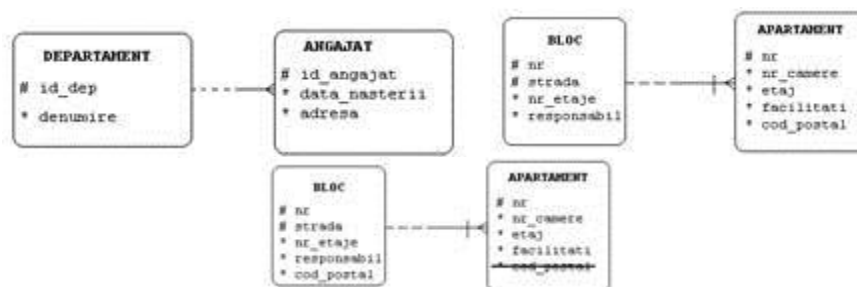
# A doua formă normală



O entitate se găsește în a doua formă normală **dacă și numai dacă se găsește în prima formă normală și în plus orice atribut care nu face parte din UID (unique identifier) va depinde de întregul UID nu doar de o parte a acestuia.**

De exemplu dacă memorăm angajații unui departament într-o entitate ca mai jos:

Se observă că **data\_nasterii** și **adresa** sunt două atribute care depind doar de **id-ul** angajatului nu de întregul UID care este combinația dintre atributele **id\_dep** și **id\_angajat**. Această situație se rezolvă prin crearea unei noi entități **ANGAJAT**, pe care o legăm de entitatea **DEPARTAMENT** printr-o relație **1 : m**.



O situație mai specială este în cazul relațiilor barate, când trebuie ținut seama că UID-ul unei entități este compus din atribute din entitatea respectivă plus un atribut sau mai multe atribute provenite din relația barată. Să considerăm următorul exemplu:

Se observă că UID-ul entității **APARTAMENT** este compus din combinația a trei atribute: numărul apartamentului, numărul blocului și strada. Deci toate atributele din entitatea **APARTAMENT** care nu fac parte din UID, trebuie să depindă de întregul UID. Dar se știe că atributul **cod\_postal** depinde doar de strada și de numărul blocului, nu și de numărul apartamentului. Acest lucru ne spune că acest atribut nu este memorat la locul potrivit. Deoarece depinde doar de combinația (strada, nr\_bloc), înseamnă că de fapt depinde de UID-ul entității **bloc**. Așadar vom muta atributul **cod\_postal** în entitatea **BLOC**.

**Observație.** Dacă o entitate se găsește în prima formă normală și UID-ul său este format dintr-un singur atribut atunci ea se găsește automat în a doua formă normală.

## A treia formă normală

O entitate se găsește în a treia formă normală dacă și numai dacă se găsește în a doua formă normală și în plus nici un atribut care nu este parte a UID-ului nu depinde de un alt atribut non-UID. Cu alte cuvinte nu se acceptă dependențe tranzitive, adică un atribut să depindă de UID în mod indirect.

Luăm ca exemplu entitatea **CARTE** din figura I.2.10. Atributul **biografie\_autor** nu depinde de **ISBN** ci de atributul **autor**. Nerezolvarea acestei situații duce la memorarea de date redundante, deoarece biografia unui autor va fi memorată pentru fiecare carte scrisă de autorul respectiv. Rezolvarea acestei situații este să creăm o nouă entitate **AUTOR**, pe care o legăm de entitatea **CARTE** printr-o relație **1 : m** (figura I.2.11.).



Figura I.2.10.

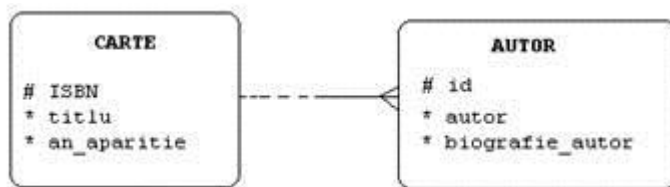


Figura I.2.11.

Atributul nu poate avea alte atribute, așa ca el devine entitate.

## 3. Relații exclusive (arce)

În unele situații, relațiile se pot exclude reciproc, adică dintr-un grup de relații, la un moment dat doar una dintre ele poate avea loc. De exemplu, un cont anume la o bancă este deținut fie de o persoană fizică fie de o firmă dar nu de ambele tipuri de clienți simultan. Un grup de relații exclusive este reprezentat în harta relațiilor printr-un arc peste relațiile care fac parte din respectivul grup, ca în figura I.4.2. Toate relațiile ce fac parte din grupul de relații exclusive trebuie să aibă aceeași opționalitate. Un arc aparține unei singure entități, adică va include doar relații care pleacă de la o aceeași entitate.

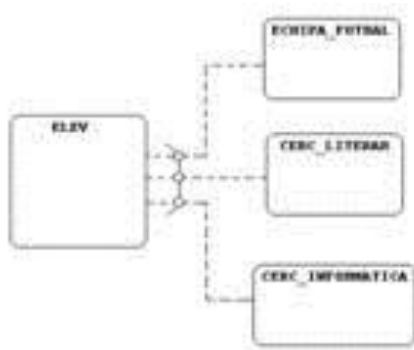
O entitate poate avea mai multe arce, dar o anumită relație nu poate face parte decât dintr-un singur arc.

Există două tipuri de relații exclusive:

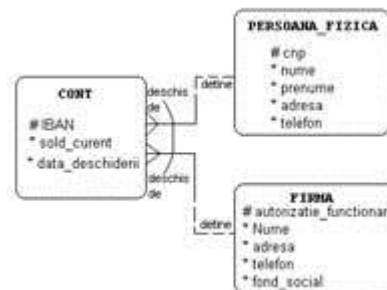
- relații exclusive **obligatorii** în care toate relațiile ce fac parte din arcul respectiv sunt obligatorii, ceea ce înseamnă că de fiecare dată, una dintre relații are obligatoriu loc. Este și cazul din figura 1 Evident că un cont trebuie să fie deținut de o persoană fizică sau de o firmă, o a treia variantă neexistând.

- relații exclusive **opționale** caz în care toate relațiile ce fac parte din arc sunt opționale. În acest caz de fiecare dată are loc cel mult una dintre relații, existând varianta ca pentru o instanță a entității căreia aparține arcul să nu aibă loc nici una din relațiile din grupul respectiv. În figura 2, este exemplificată situația în care un elev poate opta să facă parte din echipa de fotbal, sau să participe la cercul literar sau la cercul de informatică. Însă regulile școlii prevăd ca un elev să nu participe la două astfel de activități extrașcolare. Relațiile fiind opționale, înseamnă că un elev are libertatea de a decide să nu participe la nici o activitate extrașcolară.

. Relații exclusive obligatorii



Relații  
exclusive  
opționale



## Relații ierarhice. Relații recursive

Haideți să analizăm care este structura personalului într-o firmă oarecare. În figura I.1.8 este prezentată doar o parte din organigrama unei firme.



**Figura I.1.8.** Organigrama unei firme

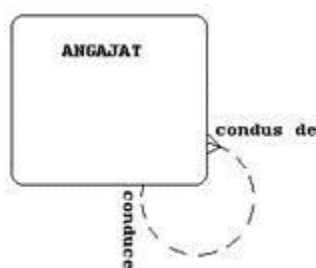
Un model de proiectare a unei astfel de structuri într-o bază de date ar fi cea din figura următoare:





**Figura I.1.9.** Implementarea unei structuri ierarhice

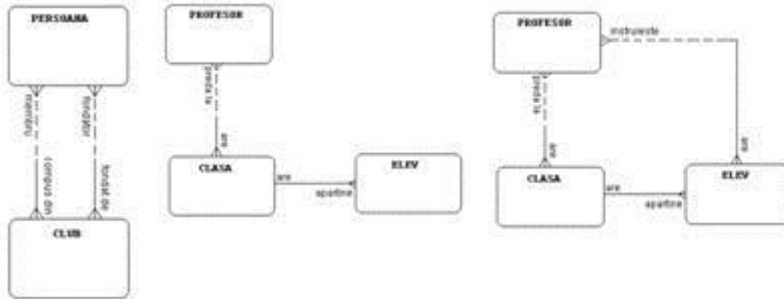
Problema este că fiecare tip de angajat din figura anterioară este de fapt un angajat și probabil există foarte multe atribute comune tuturor acestor entități ca de exemplu nume, prenume, adresă, telefon, email, data nașterii etc. Vom putea de aceea modela această structură cu ajutorul unei singure entități numită **ANGAJAT**. Însă fiecare angajat poate fi condus de către un alt angajat. Așadar vom avea o relație de la entitatea **ANGAJAT** la ea însăși. O astfel de relație se numește *relație recursivă*.



**Figura I.1.10.** Implementarea unei structuri ierarhice folosind relații recursive

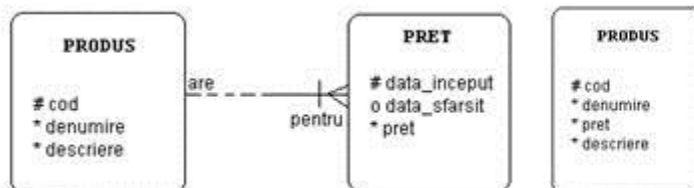
## Relații redundante si multiple

Atunci când o relație poate fi dedusă din alte relații spunem că acea relație este redundantă. Relația se poate elimina. Pot exista și relații multiple între entități



## Modelarea datelor istorice

Viața înseamnă schimbare, orice lucru se schimbă de-a lungul timpului, și nu doar obiectele se modifică în timp dar chiar și relațiile dintre aceste obiecte se schimbă. Prețul produselor poate suferi modificări destul de des. Factorii care duc la aceste modificări pot fi dintre cei mai diverși, rata inflației, anotimpul etc. Așadar atributul **preț** din cadrul entității **produs** se modifică de-a lungul timpului. Dacă nu ne interesează decât prețul actual al fiecărui produs modelul este foarte simplu, ca cel din fig. Dacă însă pentru afacerea modelată este important să reținem un istoric al prețurilor pentru fiecare produs, atunci atributul preț se va transforma într-o nouă entitate



Atributul **data\_sfarsit** este opțional, deoarece data până la care este valabil prețul curent al unui produs nu este de obicei cunoscut.

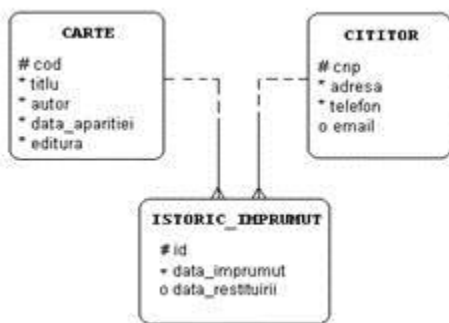
Vom considera acum o situație puțin mai dificilă. Să presupunem că dorim să modelăm o

bază de date pentru o bibliotecă. Evident este important de reținut un istoric al tuturor împrumuturilor, deoarece pe baza acestora, se pot afla domeniile de interes ale cititorilor, și astfel vom ști ce achiziții de carte să facem în viitor, vom putea determina uzura cărților astfel încât să le putem înlocui etc.

Într-o primă fază vom obține o relație de many-to-many între entitățile **CARTE** și **CITITOR**. Fiecare carte poate fi împrumutată de mai mulți cititori (evident nu în același timp), și fiecare cititor poate împrumuta mai multe cărți.



Să rezolvăm această relație many-to-many. Aplicând ceea ce am învățat în capitolele anterioare vom obține schema din fig. a 2 a



Să verificăm că acest caz este cel corect. Cheia primară este acum combinația coloanelor **cod\_carte** și **data\_imprumut**. Poate un cititor împrumuta două cărți în aceeași dată? Adică următoarele două înregistrări pot exista simultan în tabela **ISTORIC\_IMPRUMUTURI**? Răspunsul este DA, combinația celor două coloane, pentru cele două înregistrări fiind unică.

Deci bararea automată a celor două relații dinspre entitatea de intersecție nu este întotdeauna o soluție corectă. Pentru a evita aceste complicații putem recurge la introducerea unei chei artificiale în entitatea de intersecție. În exemplul nostru se poate decide ca pentru fiecare împrumut în parte să se completeze câte o fișă separată care are un număr unic. Obținem modelul din figura I.4.13, care este de asemenea unul corect.

**Fig.** Introducerea unei chei artificiale

**Convenții de ridabilitate:** Se aplica convențiile Oracle de scriere a ERD-ului:

Entitatea se scrie cu majuscule, singular în interiorul unui dreptunghi cu vf rotunjite. Atributele se scriu cu litere mici, având în față unul din semnele #, \*, o (UID, obligatoriu, optional). Orientarea liniilor este de la V la E și de sus în jos, evitând intersecția. Se pot folosi subdiagrame de explicare a diagramelor complexe, și explicarea entitatilor cu multe atribute.

## Modelarea generica

Modelul generic aduce beneficii dacă cerințele afacerii se schimbă des. Atunci e nevoie de entități și atribute noi. Se poate modela o singură entitate **Article type** care să păstreze oricâte tipuri de articole e nevoie, aceasta reduce nr de entități.

## Procesul mapării

Transformarea modelului conceptual, a ERD-ului, în modelul fizic, adică în baza de date propriu zisă, se numește mapare. Acest proces implică transformarea fiecărui element al ERD-ului.

Entități → tabele, (CARTE-carti.dbf)

atribute → câmpuri,

coloane, UID → cheie primară,

relație → cheie străină,

business rules → constrângeri

Se mapează procesul de transformare în diagrama tabeli:

Tipuri de date in Oracle:

Numele coloanei	Tip	Tip cheie	Opționalitatea
titlu	Varchar2	Pk	*
autor	Varchar2	Pk	*
data_apariției	Date		*
Format	Varchar2		*
Nr_pagini	Number		*

Tipul de date	Descriere	Dimensiune Maximă
<b>VARCHAR2</b>	Șir de caractere de lungime variabilă	<b>4000</b> bytes
<b>CHAR</b>	Șir de caractere de lungime fixă	<b>2000</b> bytes
<b>NUMBER (p , s)</b>	Număr având p cifre din care s la partea zecimală. (s negativ reprezintă numărul de cifre semnificative din fața punctului zecimal)	p (precizia) între 1 și 38. s (scala) între -84 și 127.
<b>DATE</b>	Data calendaristică	De la 1 Ianuarie 4712 BC pana la 31 Decembrie, <b>9999</b> AD.
<b>TIMESTAMP</b>	Se memorează data calendaristică, ora, minutul, secunda și fracțiunea de secundă	Fracțiunea de secundă este memorată cu o precizie de la 0 la 9.
<b>INTERVAL YEAR TO MONTH</b>	perioadă de timp în ani și luni.	
<b>INTERVAL DAY TO SECOND</b>	memorează un interval de timp în zile, ore, minute și secunde	
<b>CLOB</b>	Character Large Object	4 Gigabytes

<b>BLOB</b>	Binary Large Object	4 Gigabytes
<b>BFILE</b>	Se memorează adresa unui fișier binar de pe disc	4 Gigabytes

dacă relația pe partea many este opțională atunci și coloanele cheii străine vor fi opționale. Ce înseamnă acest lucru? Faptul că un jucător poate la un moment dat să nu joace la nici o echipă, atunci câmpul `cod echipă` va rămâne necompletat în dreptul lui (va avea valoarea NULL). Dacă însă relația este obligatorie pe partea many atunci coloanele ce fac parte din cheia străină vor fi opționale.

În general, la maparea unei relații de tip one-to-many, vom introduce în tabela corespunzătoare entității de pe partea many a relației cheia primară a entității de pe partea one a relației. Câmpurile astfel introduse se vor numi **cheie străină** (foreign key).

Așadar:

- cheia străină a unei tabele este cheia primară din tabela referintă
- cheia străină este întotdeauna introdusă în tabela corespunzătoare entității din partea many a relației.

## Maparea relațiilor one-to-one

Dându-se două entități **A** și **B** legate între ele printr-o relație one-to-one, este evident că putem include cheia primară **A** în cadrulul tablei **B**, dar putem proceda la fel de bine și invers, incluzând cheia primară a tablei **B** în cadrul tablei **A**, deoarece fiecărei instanțe a entității **A** îi corespunde cel mult o instanță a entității **B**, dar și invers, oricărei instanțe a entității **B** îi corespunde cel mult o instanță a entității **A**.

Pentru relația din figura I.3.3 de exemplu putem memora pentru fiecare persoană seria de pașaport, dar și invers, pentru fiecare pașaport putem memora cnp-ul deținătorului.

**Figura I.3.3.**

Decizia depinde de specificul afacerii modelate. Dacă de exemplu ne interesează în primul rând persoanele și abia apoi datele de pe pașapoarte, atunci vom adopta probabil prima variantă, a memorării seriei de pașaport în cadrul tablei **PERSOANE**, dacă însă baza de date este destinată evidenței pașapoartelor, atunci probabil vom adopta varianta a doua.

Uneori este convenabil să memorăm cheia străină în ambele părți ale relației, în exemplul nostru pentru fiecare pașaport să memorăm cnp-ul persoanei care îl deține, dar și pentru fiecare persoană să memorăm seria de pașaport.

## Maparea relațiilor recursive

Dacă vom privi o relație recursivă ca pe o relație de tipul one-to-many între o entitate și ea însăși, atunci acest caz se reduce la ceea ce deja am discutat. Să exemplificăm relația din figura I.3.4. Relația recursivă din această figură poate fi privită ca o relație între două entități identice, ca în figura I.3.5.

**Figura I.3.4.**

**Figura I.3.5.**

Așadar vom introduce în cadrul tabelului **ANGAJAȚI**, marca șefului său. Diagrama de tabelă va arăta ca mai jos.

**Tabelul I.3.4.**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Marca	Number	Pk	*
Nume	Varchar2		*
Prenume	Varchar2		*
Data_angajarii	Date		*
Adresa	Varchar2		*
Telefon	Varchar2		o
Email	Varchar2		o
Marca_sef	Number	Fk	o

## I.3.4. Maparea relațiilor barate

Relațiile barate sunt mapate ca cheie străină în tabela aflată în partea many a relației, la fel ca la maparea oricărei relații one-to-many. Bara de pe relație exprimă faptul că acele coloane ce fac

parte din cheia străină vor devenii parte a cheii primare a tabelului din partea many a relației barate.

Pentru exemplul din figura I.3.6, cheia primară a tabelului **ATTRIBUTE** va fi format din coloanele **denumire\_atribut** și **denumire\_entitate**, aceasta din urmă fiind de fapt cheie străină în tabela **ATTRIBUTE**.

**Figura I.3.6.** Maparea relațiilor barate

**Tabelul I.3.5.** Tabela **ENTITĂȚI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
denumire	Varchar2	Pk	*

**Tabelul I.3.5.** Tabela **ATTRIBUTE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
denumire_atribut	Varchar2	Pk	*
denumire_entitate	Varchar2	<b>Pk, Fk</b>	*
optionalitate	Varchar2		*

Să considerăm acum un exemplu în care există mai multe relații barate, în cascadă.

**Figura I.3.7.** Relații barate în cascadă

**Tabelul I.3.6.** Tabela **A**

Numele coloanei	Tip cheie	Opționalitate
<b>idA</b>	<b>Pk</b>	*
C1		*

**Tabelul I.3.7.** Tabela **B**

Numele coloanei	Tip cheie	Opționalitate
<b>idB</b>	<b>Pk</b>	*
C2		*
<b>idA</b>	<b>Pk, Fk</b>	*

**Tabelul I.3.8.** Tabela **C**

Numele coloanei	Tip cheie	Opționalitate
-----------------	-----------	---------------

**Tabelul I.3.9.**  
Tabela **D**

<b>idC</b>	<b>Pk</b>	*
C3		*
<b>idA</b>	<b>Pk, Fk</b>	*
<b>idB</b>	<b>Pk, Fk</b>	*

Numele coloanei	Tip cheie	Opționalitatea
<b>idD</b>	<b>Pk</b>	*
C4		*
<b>idA</b>	<b>Fk</b>	*

## . Operații specifice prelucrării bazelor de date

Orice sistem de gestiune a bazelor de date (SGBD) trebuie să asigure următoarele **funcții**:

- definirea structurii bazei de date
- încărcarea datelor în baza de date (adăugarea de noi înregistrări la baza de date)
- accesul la date pentru:
  - interogare (afișarea datelor, sortarea lor, calcule statistice etc.)
  - ștergere
  - modificare
- întreținerea bazei de date:
  - refacerea bazei de date prin existența unor copii de siguranță
  - repararea în caz de incident
  - colectarea și re folosirea spațiilor goale
- posibilitatea de reorganizare a bazei de date prin:
  - restructurarea datelor
  - modificarea accesului la date
- securitatea datelor.

O parte din aceste operații pot fi realizate cu ajutorul limbajului SQL, altele cu ajutorul unor programe specializate, care sunt puse la dispoziția administratorului bazei de date de către sistemul de gestiune al bazelor de date.



# . Reguli de integritate

Detalierea caracteristicilor pe care trebuie să le prezinte un SGBD pentru a fi considerat relațional s-a făcut de E. F. Codd în 1985 sub forma a 13 reguli. Una dintre aceste reguli precizează că restricțiile de integritate trebuie să poată fi definite în limbajul utilizat de SGBD pentru definirea datelor.

Regulile de integritate garantează că datele introduse în baza de date sunt corecte și valide. Aceasta înseamnă că dacă există orice o regulă sau restricție asupra unei entități, atunci datele introduse în baza de date respectă aceste restricții. În Oracle, regulile de integritate se definesc la crearea tabelelor folosind **constrângerile**. Dar asupra acestora vom reveni în partea a doua a manualului.

Tipurile de reguli de integritate sunt următoarele:

- **Integritatea entităților** – indică faptul că nici o coloană ce face parte din cheia primară nu poate avea valoarea NULL. În plus, pentru fiecare înregistrare, cheia primară trebuie să fie unică.
- **Integritatea de domeniu** – acest tip de reguli permite ca într-o anumită coloană se introducă doar valori dintr-un anumit domeniu. De exemplu putem impune ca salariul unui angajat să fie cuprins între 4500 și 5000 RON.
- **Integritatea referențială** – este o protecție care asigură ca fiecare valoare a cheii străine să corespundă unei valori a cheii primare din tabela referită. De exemplu, referindu-ne la tabelele **JUCĂTORI** și **ECHIPE**, corespunzătoare ERD-ului din figura I.3.2, **cod** este cheie primară în tabela **ECHIPE**, iar în tabela **JUCĂTORI**, **cod** devine cheie străină. Astfel valoarea câmpului **cod** din cadrul tablei **JUCĂTORI** corespunzătoare unui anumit jucător trebuie să se regăsească printre valorile câmpului **cod** din tabela **ECHIPE**, altfel ar însemna că jucătorul respectiv joacă la o echipă inexistentă (vezi figura I.3.8).

**Figura I.3.8.** Exemplu de încălcare a integrității referențiale

Situații de încălcare a integrității referențiale pot apărea:

- la **adăugarea** unei noi înregistrări în baza de date, se poate încerca introducerea unor valori invalide pentru câmpurile cheii străine;
- la **actualizarea** bazei de date;

- la **ștergerea** unei înregistrări. De exemplu se șterge înregistrarea corespunzătoare unei anumite echipe (echipa se desființează). Înregistrările jucătorilor care au jucat la acea echipă vor încălca integritatea referențială, deoarece se vor referi la o echipă care nu mai există. Soluțiile posibile sunt ca la ștergerea unei echipe, toți jucătorii care au activat la acea echipă să fie și ei șterși din baza de date (ștergere în cascadă) sau valoarea câmpului **cod echipă** pentru acei jucători să fie setată la **NULL**, ceea ce va înseamnă că acei jucători nu activează la nici o echipă.

## Programe de validare și de acțiune

În realizarea modelului conceptual al unei baze de date se ține cont de modul în care funcționează afacerea modelată, datele care trebuie să fie memorate, relațiile dintre acestea etc. Modul de utilizare a diferitelor date, modul în care acestea sunt relaționate pot diferi de la o afacere la alta.

Regulile afacerii unei organizații se referă în esență la procesele și fluxurile tuturor datelor și activităților zilnice din cadrul organizației. Cum funcționează organizația? Care sunt activitățile sale?

Regulile afacerii acoperă următoarele aspecte ale unei organizații:

- Orice tip de politici organizaționale de orice tip și de la orice nivel al organizației.
- Orice tip de formule de calcule (ca de exemplu modul de calcul al ratelor pentru diverse împrumuturi, modul de calcul al salariilor etc)
- Orice tip de reguli impuse de lege sau reguli interne ale organizației.

Regulile simple ale afacerii pot fi implementate în modelul bazei de date prin intermediul relațiilor dintre entități. Acest tip de reguli se numesc **reguli structurale**.

Alte reguli ale afacerii pot fi implementate folosind regulile de integritate despre care am discutat în paragraful anterior. Există totuși reguli pentru implementarea cărora va trebui să scriem programe speciale folosind limbaje specializate specifice SGBD-ului utilizat. Acest tip de reguli se numesc numite **reguli procedurale**. În Oracle acest tip de programe se vor scrie folosind limbajul PL/SQL (Procedural Language/Structured Query Language) și se numesc **declanșatoare (triggere)**.

Există două tipuri de declanșatoare:

- declanșatoare de aplicație care se execută când apar anumite evenimente la nivelul anumitor evenimente;

- declanșatoare ale bazei de date care sunt lansate în execuție când apar diverse evenimente asupra datelor (de exemplu la executarea unor comenzi ca **INSERT**, **UPDATE**, **DELETE**) sau la apariția unor evenimente system (logarea la baza de date sau delogarea).

Orice declanșator poate avea rol de validare a unei operații, poate realiza diferite operații suplimentare, ca de exemplu diferite calcule, caz în care vom spune că e vorba de un declanșator de acțiune.

## Maparea tipurilor și subtipurilor

Nici un sistem de gestiune a bazelor de date nu suportă în mod direct supertipurile și subtipurile. Putem adopta mai multe soluții ale acestei probleme. Vom exemplifica aceste variante pentru schema din figura I.4.1, în care, pentru simplitate, vom presupune că nu avem nevoie de subentitatea **ALTUL**.

**Varianta 1.** Vom crea o tabelă pentru supertip și câte o tabelă pentru fiecare subtip. Diagramele de tabelă în acest caz vor fi:

**Tabelul I.4.1.** Tabela **ANGAJAȚI SECRETARE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Nume	Varchar 2		*
Adresa	Varchar 2		*
Data_nasterii	Date		*
Id_departament	Number	Fk	*

**Tabelul I.4.2.** Tabela

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*

**Tabelul I.4.3. Tabela **MANAGERI**  
REPREZENTANȚI\_VÂNZĂRI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*

**Tabelul I.4.4. Tabela**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Zona_vanzari	Varchar2		*
Permis_conducere	Varchar2		*

Bonus	Number		*
Id_depart_conduc s	Number	Fk	o

Am notat cu **Id\_depart\_conduc** codul departamentului pe care îl conduce un manager, iar cu **Id\_departament** codul departamentului în care lucrează un anumit angajat.

Cheia primară a supertipului va fi inclusă în toate tabelele corespunzătoare subtipurilor și va deveni cheia primară a acelei tabele.

Atributele și cheile străine provenite din relațiile de la nivelul supertipului vor fi memorate în tabela corespunzătoare supertipului. Atributele și relațiile de la nivel de subtip, se vor memora doar în tabela corespunzătoare subtipului respectiv.

Acest model este cel mai natural dar poate crea multe probleme privind eficiența întrucât sunt necesare multe operații de interogare din tabele multiple, pentru a obține informații suplimentare despre toți angajații.

**Varianta 2.** Vom crea câte o tabelă pentru fiecare subtip. Atributele și cheile străine provenite din relațiile de la nivelul supertipului vor fi introduse în fiecare tabelă astfel obținută, acestea fiind moștenite de către fiecare subtip.

**Tabelul I.4.5. Tabela SECRETARE**Tabela **MANAGERI**

Numele coloanei	Tip	Tip cheie	Opționalitate
Id_angajat	Number	Pk	*
Nume	Varchar 2		*
Adresa	Varchar 2		*
Id_departament	Number	Fk	*
Data_nasterii	Date		*

**Tabelul I.4.6.**

Numele coloanei	Tip	Tip cheie	Opționalitate
Id_angajat	Number	Pk	*
Nume	Varchar2		*
Adresa	Varchar2		*
Data_nasterii	Date		*
Bonus	Number		*
Id_depart_condu s	Number	Fk	o
Id_departament	Number	Fk	*

**Tabelul I.4.7. Tabela REPREZENTANȚI\_VÂNZĂRI**

Numele coloanei	Tip	Tip cheie	Opționalitate
Id_angajat	Number	Pk	*
Nume	Varchar 2		*
Adresa	Varchar 2		*
Data_nasterii	Date		*
Id_departament	Number	Fk	*
Zona_vanzari	Varchar 2		*
Permis_conducere	Varchar 2		*

**Varianta 3.** Vom crea o singură tabelă pentru supertip. Această tabelă va conține toate coloanele corespunzătoare atributelor de la nivelul supertipului, dar și toate coloanele corespunzătoare tuturor atributelor din toate subtipurile. Atributele de la nivelul supertipului își vor păstra opționalitatea, însă atributele de la nivelul subtipurilor, vor fi toate introduse în tabelă, dar vor fi toate opționale.

Relațiile de la nivelul supertipului se transformă normal. Relațiile de la nivelul subtipurilor se vor implementa cu ajutorul cheilor străine opționale.

**Tabelul I.4.8. Tabela ANGAJAȚI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Nume	Varchar 2		*
Adresa	Varchar 2		*
Id_departament	Number	Fk	*
Data_nasterii	Date		*
Bonus	Number		o
Id_depart_conduc s	Number	Fk	o
Zona_vanzari	Varchar 2		o
Permis_conducere	Varchar 2		o
Tip_angajat	Numeri c		*

Am introdus un atribut suplimentar **Tip\_angajat**, cu ajutorul căruia vom codifica dacă un angajat este manager, secretară sau reprezentant de vânzări. Deoarece attributele de la nivelul subtipurilor sunt obligatorii pentru subtipul respectiv, va trebui să stabilim o regulă de integritate la nivel de înregistrare, care să verifice că pentru o înregistrare de un tip anume sunt completate câmpurile corespunzătoare. De exemplu, la adăugarea unui nou manager în tabela **ANGAJAȚI**, trebuie să verificăm dacă este completat câmpul bonus.

Se observă că vor fi multe câmpuri cu valoarea null, ceea ce înseamnă o risipă de spațiu de memorie.

**Tabelul I.4.9. Tabela ANGAJAȚI**

Id_angajat	Bonus	Id_departament_conduc	Zona_vanzari	Permis_conducere	Tip_angajat	...
10	125	5	(null)	(null)	1	
121	(null)	(null)	Transilvania	568147	2	
245	(null)	(null)	(null)	(null)	3	
...						

În acest tabel am codificat managerii cu 1, reprezentanții de vânzări cu 2, iar secretarele cu 3. Așadar această variantă de implementare este convenabilă când există puține atribute și relații la nivelul subtipurilor.

# . Maparea arcelor

Pentru a mapa un arc vom crea atâtea chei străine câte relații există în arcul respectiv. Pentru modelul din figura I.4.2 vom obține următoarele tabele:

**Tabelul I.4.10.** Tabela **CONTURI**  
**PERSOANE\_FIZICE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
-----------------	-----	-----------	----------------

**Tabelul I.4.11.** Tabela

Numele coloanei	Tip	Tip cheie	Opționalitatea
Cnp	Number	Pk	*
Nume	Varchar 2		*
Prenume	Varchar 2		*
Adresa	Varchar 2		*
Telefon	Number		*

IBAN	Number	Pk	*
Sold_curent	Number		*
Data_deschiderii	Date		*
Cnp	Number	Fk1	o
Autorizatie_function are	Number	Fk2	o

Numele coloanei	Tip	Tip cheie	Opționalitatea
Autorizatie_function are	Number	Pk	*
Nume	Varchar2		*
Adresa	Varchar2		*
Telefon	Number		*
Fond_social	Number		*

**Tabelul I.4.12.** Tabela **FIRME** Deși relațiile din arc sunt obligatorii, cheile străine corespunzătoare au fost setate ca fiind opționale, deoarece pentru fiecare înregistrare trebuie să avem completată una din cele două chei străine, iar cealaltă cheie străină trebuie să rămână necompletată (principiul exclusivității). Va trebui să implementăm o condiție de integritate care să verifice această condiție.